An example of using JASMIN with R

**Step 1:**

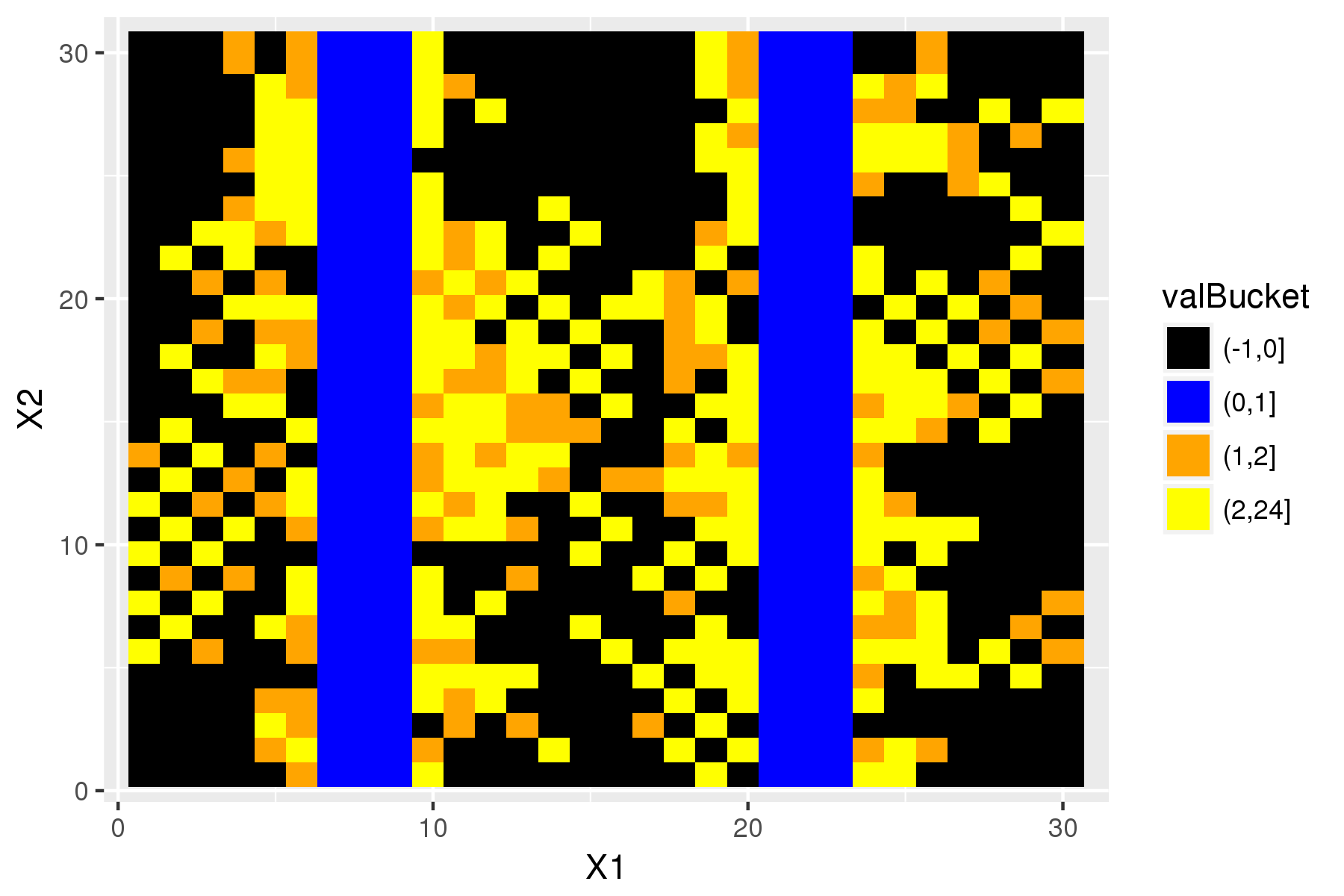
**Look at the original script ‘original\_script.R’**

We take an area (it could be a country, state, city) that is made up of units (cells) that can be settled (occupied). We allow the initial ‘pioneers’ to settle first (Seeding the cells). We then establish some rules for future residents to settle (Settling Rules). Once we have run the simulation we plot the area, and collect some stats to study the growth of the population under different starting conditions and different settling rules.

There are a range of different parameters that we can change, and each will affect the results. We want to study the effect of four parameters:

* kNumSettlers – The number of settlers who follow the pioneers
* kMaxLookAroundSteps - How many scouting attempts will settlers make, before abandoning
* seeding.opt – How are pioneers seeded? 1: Random, 2: Central Square, 3: Central Ring, 4: Two Columns.
* settling.option - How are settlers seeded? 1: Random, 2: NSEW only, 3: Diagonal settling only.

Run the script and see what it produces. Try changing some of the parameter values above and see what changes.



**Step 2:**

**Make this a loop that we can run through all combinations of the parameters**

We have been given a file (data/parameters.csv) which specifies all of the combinations of the four parameters that need testing, a total of 300 simulations. We need to re-write the R script to loop through the rows of this table, running all the simulations required. We also need to adapt the code so that the plot is saved as a ‘.png’ and the summary table is saved as a ‘.csv’.

This has been done in the R script provided, ‘loop\_parameters\_file.R’.

The loop uses an index, ‘i’, to pull out the correct parameters from the parameters table and saves the table and the plot to 2 folders, one for plots and one for tables.

This example takes ~30 minutes to run locally, but could be much longer if we increased the number of iterations or the number of parameters we varied. A 30 minute job is not long enough to benefit from being run on JASMIN (it might take about 20 minutes to get it set up on JASMIN), but as an example it is useful.

**Step 3:**

**Move all the files to JASMIN**

Use WinSCP or MobaXterm to move all of the files over to your workspace on JASMIN. Once moved login in to JASMIN and navigate to these files in your working directory.

**Step 4:**

**Have a look at the job file and JASMIN script**

The job file is vital to running your code on JASMIN, it has all the important information such as which queue you job will go to, how much time you need, etc.

One thing to note in this job file is the notation of the job: “-J Pop\_Sim[1-300]”. This means the job has the name ‘Pop\_Sim’ and will be run 300 times, first with the index 1, then 2, and so on to 300.

Now have a look the R scipt that has been written for JASMIN, ‘JASMIN\_parameters\_file.R’. This is almost identical to the script in step 2. The only change is that the ‘for’ loop has been replaced with the line:

i <- as.numeric(commandArgs(trailingOnly = TRUE))[1]

This line takes the index value from the job file (1, 2, …, 300) and then proceeds as in the for loop to get the required row from the parameters file. We use the parameters file to parameterise our simulation however you may use it differently in your own work. For example you might instead use a list of species names (running a model for each) or a list of file names (to reformat or analyse).

**Step 5:**

**Install R packages**

The scripts we are going to use require 3 R packages that you wont have installed. These can be installed in exactly the same way as on your local PC. Type ‘R’ at the console and an R console will launch. From here use the function ‘install.packages’ to install ‘ggplot2’, ‘plyr’ and ‘reshape’.

**Step 6:**

**Run our job**

Now that we have our R script ready and the job script written we can run our job. First make sure you are in the directory with the pop\_sim.job file. When here the job can be submitted with ‘bsub < pop\_sim.job’

If the job is submitted successfully you will then get a message like:

Job <XXXXXXXX> is submitted to queue <short-serial>

You can monitor the progress of your job by typing ‘busers’. This command shows a summary of you activity, with columns for the total number of jobs you have on JASMIN (‘njobs’), and how many of these are pending (‘pend’), versus running (‘run’).

If things go well then output will appear in the two output folders. If things go wrong and output does not appear or looks wrong the console folder contains all the console outputs from R, it is here that you will find errors messages that will help with debugging.